Cuando escribimos algo en la pantalla, lo que hacemos es alterar la situación de los puntos de la celda de carácter afectada. Es menos obvio, pero cierto, que también podemos cambiar los atributos de la celda. Mientras no especifiquemos otra cosa, todo se escribe con tinta negra sobre papel blanco (con brillo normal y sin parpadeo); sin embargo, podemos cambiar esta situación utilizando las sentencias INK, PAPER, BRIGHT y FLASH. Seleccione la opción Pantalla del menú de edición para llevar el cursor a la pantalla inferior; dé la siguiente orden:

```
PAPER 5
```

y luego escriba (con PRINT) unos cuantos caracteres en la pantalla; aparecerán sobre papel cyan, porque éste es el color que habíamos elegido para el papel (el código del color cyan es el 5).

Las otras instrucciones funcionan de manera similar, así que con

```
PAPER   (número entero entre 0 y 7)
INK     (número entero entre 0 y 7 )
BRIGHT  (número entero entre 0 y 1)
FLASH   (número entero entre 0 y 1)
```

podemos controlar todos los atributos de los caracteres que escribamos a continuación.

Pruebe todas estas instrucciones. Cuando lo haya hecho ya podrá entender cómo funcionaba el programa del principio de esta sección (recuerde que un espacio es un carácter en el que todos los puntos tienen el color del papel).

Hay otros números que podemos usar en estas sentencias y cuyos efectos son menos directos.

El 8 se puede usar en todas las sentencias y significa 'transparente', en el sentido de que respeta el atributo que estuviera antes en vigor. Supongamos, por ejemplo, que damos la orden

```
PAPER 8
```

Ninguna posición de carácter puede tener este color de papel, sencillamente porque no existe el color 8; lo que ocurre es que cuando escribamos sobre una posición, el color del papel seguirá siendo el que hubiera antes en ella. Sin embargo, INK 8, BRIGHT 8 y FLASH 8 funcionan del mismo modo que los otros números de atributo.

El número 9 sólo es aplicable a PAPER e INK y significa 'contraste'. Sirve para hacer que el color (de la tinta o del papel) utilizado contraste con el otro, convirtiéndolo en blanco si el otro es un color oscuro (negro, rojo o magenta) o haciéndolo negro si el otro es claro (verde, cyan, amarillo o blanco).

Compruébelo dando estas instrucciones:

```
INK 9: FOR c=0 TO 7: PAPER c: PRINT c: NEXT c
```

Una muestra más impresionante de su potencia es ejecutar el programa del principio para que dibuje las rayas de colores (de nuevo, seleccione la pantalla inferior antes de escribir RUN) y luego hacer lo siguiente:

```
INK 9: PAPER 8: PRINT AT 0,0;: FOR n=1 TO 1000: PRINT n;: NEXT n
```

El color de la tinta contrasta siempre con el color antiguo del papel en todas las celdas de carácter.

Los televisores de color en realidad sólo utilizan tres colores: rojo, verde y azul. Todos los demás colores son mezclas de éstos. Por ejemplo, el magenta se forma mezclando rojo con azul; ésta es la razón por la que su código es 3, ya que este número es la suma de los códigos del rojo (2) y el azul (1).

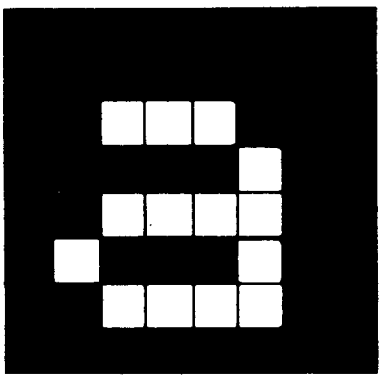
Para comprender cómo se forman los ocho colores, imagine tres focos rectangulares, de colores rojo, verde y azul, que arrojan su luz en la oscuridad sobre tres zonas parcialmente solapadas de un pedazo de papel blanco. Donde las zonas se solapen verá usted mezclas de colores, como muestra el programa siguiente (observe que los cuadrados de tinta sólida se obtienen activando el modo gráfico con la tecla [GRAF] y pulsando luego la tecla del 8 en combinación con [MAYUSC]; para salir del modo gráfico se pulsa la tecla del 9):

```
 10 BORDER 0: PAPER 0: INK 7: CLS
 20 FOR a=1 TO 6
 30 PRINT TAB 6; INK 1;"                ": REM 18 cuadrados de tinta
 40 NEXT a
 50 LET línea datos=200
 60 GO SUB 1000
 70 LET línea datos=210
 80 GO SUB 1000
 90 STOP
200 DATA 2,3,7,5,4
210 DATA 2,2,6,4,4
1000 FOR a=1 TO 6
1010 RESTORE línea datos
1020 FOR b=1 TO 5
1030 READ c: PRINT INK c;"     ";: REM 6 cuadrados de tinta
1040 NEXT b: PRINT: NEXT a
1050 RETURN
```

Hay una función, llamada ATTR, que averigua los atributos de una posición dada de la pantalla. Es una función complicada, y por eso la aplazaremos para el final de esta sección.

Disponemos de otras dos sentencias, INVERSE y OVER, que no controlan los atributos, sino la distribución de puntos del carácter que resulta impreso en la pantalla. Llevan como parámetro el número 0, que significa 'activado', o el número 1, que significa

'desactivado'. Así, INVERSE 1 invierte la situación de todos los puntos de la retícula: apaga los encendidos y enciende los apagados. Por ejemplo, la celda de carácter de la letra 'a' se convierte en la que se muestra en la siguiente figura:



Si, como ocurre en el momento de encender el ordenador, tenemos tinta negra y papel blanco, la 'a' aparecerá en blanco sobre negro.

La sentencia

```
OVER 1
```

activa un tipo particular de sobreimpresión. Normalmente, cuando se escribe algo en una posición de la pantalla, el carácter nuevo «tapa» completamente lo que hubiera antes en ella; en cambio, si previamente se da la orden OVER 1, el nuevo carácter se funde con el antiguo. Esto permite escribir caracteres compuestos, por ejemplo, letras subrayadas, como en el programa siguiente. (Reinicialice el +2 y seleccione 128 BASIC. El carácter de subrayado se obtiene con SIMB y 0.)

```
10 OVER 1
20 PRINT "w"; CHR$ 8;"_";
```

(Observe que hemos usado el carácter de control CHR$ 8 para hacer retroceder la posición de escritura antes de escribir el '_'.)

Hay otra forma de usar INK, PAPER, etc. que usted probablemente encontrará más cómoda que la que hemos visto hasta ahora. Podemos utilizarlas como elementos de PRINT seguidas del signo de punto y coma, y harán exactamente lo mismo que si las hubiésemos ejecutado como sentencias independientes, con la única diferencia de que sus efectos son sólo temporales, pues duran solamente hasta el final de la sentencia PRINT que las contiene. Por tanto, si escribimos

```
PRINT PAPER 6;"x";: PRINT "y"
```

sólo la x quedará escrita sobre papel amarillo.

Cuando usamos INK y sus compañeras como sentencias independientes, no afectan al color de la pantalla inferior, en la que se realiza la captación de datos por las sentencias INPUT y donde el +2 muestra los mensajes de error. En esta parte de la pantalla, el color del papel es el del borde, el de la tinta es el 9 (contraste), carece de parpadeo y su brillo es normal. Como color del borde se puede elegir cualquiera de los ocho colores normales (no el 8 ni el 9) mediante la instrucción

```
BORDER color
```

Cuando se está ejecutando INPUT, los caracteres introducidos aparecen en tinta de contraste sobre papel del mismo color que el borde; pero se puede cambiar el color de los mensajes inductores escritos por el +2 usando para ello INK y PAPER (y las demás) como elementos de INPUT, de la misma forma que se haría en una sentencia PRINT. Sus efectos duran hasta el final de la sentencia o hasta que se termine de introducir los datos, lo que antes ocurra. Pruebe lo siguiente:

```
INPUT FLASH 1; INK 1;"¿Cual es su numero?";n
```

El +2 tiene en gran estima su salud mental: cualquiera que sea la combinación de colores producida por el programa, el editor siempre funciona con tinta negra sobre papel blanco.

Otra forma de cambiar los colores consiste en usar caracteres de control, de forma parecida a como hacíamos con AT y TAB de la Parte 15.

```
CHR$ 16  corresponde a  INK
CHR$ 17  corresponde a  PAPER
CHR$ 18  corresponde a  FLASH
CHR$ 19  corresponde a  BRIGHT
CHR$ 20  corresponde a  INVERSE
CHR$ 21  corresponde a  OVER
```

Estos caracteres de control van seguidos de un carácter que especifica un color; así, por ejemplo,

```
PRINT CHR$ 16+CHR$ 9;"elemento"
```

produce el mismo efecto que

```
PRINT INK 9;"elemento"
```

En la práctica, no tiene ningún interés utilizar estos caracteres de control, pues siempre podemos emplear en su lugar las sentencias INK, PAPER, etc. Sin embargo, si usted tiene en cassette algún programa escrito para el antiguo 48 BASIC, puede encontrar tales caracteres de control inmersos en el listado. En general, el editor los ignorará activamente y los suprimirá a la primera oportunidad. No es posible introducirlos en los listados, como se hacía en el antiguo Spectrum de 48K.

La función ATTR tiene la forma

ATTR (*fila,columna*)

Sus dos argumentos son los números de fila y columna que ya sabemos utilizar en la cláusula AT. El resultado de la función es un número que informa sobre los colores y los demás atributos de la posición de carácter especificada. Esta función puede intervenir en expresiones, con la misma libertad que cualquier otra función.

El número resultante es una combinación de cuatro números que se construye de la siguiente forma:

sumar	128	si la celda de carácter está parpadeando; 0 en otro caso
sumar	64	si la celda de carácter es brillante; 0 si es normal
sumar	8	multiplicado por el código del color del papel
sumar	1	multiplicado por el código del color de la tinta

Por ejemplo, si la celda de carácter está parpadeando, tiene un brillo normal, papel amarillo y tinta azul, los cuatro números que debemos sumar son 128, 0, 8*6=48 y 1, lo que da un total de 177. Compruébelo escribiendo:

```
PRINT AT 0,0; FLASH 1; PAPER 6; INK 1;" ";ATTR (0,0)
```

Ejercicios

1. Pruebe la siguiente orden:

   ```
   PRINT "-"; CHR$ 8; OVER 1;"/";
   ```

   En el lugar de intersección de la barra con el signo menos ha quedado un punto blanco. Ésta es la forma de sobreimpresión en el +2: papel más papel y tinta más tinta dan papel; tinta más papel da tinta. Este sistema tiene la interesante propiedad de que si sobreescribimos la misma cosa dos veces, se restaura el carácter original. Sabido esto, si ahora hacemos

   ```
   PRINT CHR$ 8; OVER 1;"/"
   ```

   ¿por qué recuperamos un '-' incólume?

2. Ejecute este programa:

   ```
   10 POKE 225227+RND*704, RND*127
   20 GO TO 10
   ```

(No se preocupe por cómo funciona el programa.) El programa está cambiando los colores de las células de la pantalla del televisor; RND garantiza que esto ocurre de forma aleatoria. Las rayas diagonales que finalmente llegan a aparecer son la manifestación del hecho de que RND no genera verdaderos números aleatorios, sino pseudoaleatorios.

Parte 17
Gráficos

Temas tratados:

PLOT, DRAW, CIRCLE
pixels

A lo largo de toda esta sección, escriba los programas de ejemplo, las órdenes y la palabra RUN en la pantalla inferior (selecciónela con la opción Pantalla del menú de edición).

En esta sección vamos a aprender a dibujar figuras en el +2. La parte de la pantalla que usted puede utilizar para los dibujos tiene 22 filas y 32 columnas, lo que da un total de 22*32=704 posiciones de carácter. Como recordará de la Parte 16, cada posición de carácter es una retícula de 8x8 puntos, llamados *pixels* (de *picture elements*, 'elementos de imagen').

Un pixel se especifica dando dos números: sus *coordenadas*. El primero, su coordenada *x*, indica a qué distancia se encuentra el pixel del borde izquierdo de la pantalla. El segundo, su coordenada *y*, indica la distancia que hay del pixel al borde inferior. Estas coordenadas se escriben generalmente entre paréntesis y separadas por comas, de modo que, por ejemplo, (0,0), (255,0), (0,175) y (255,175) representan, respectivamente, los rincones inferior izquierdo, inferior derecho, superior izquierdo y superior derecho de la pantalla.

Si no consigue recordar cuál es la coordenada *x* y cuál la coordenada *y*, quizá le sirva de ayuda asociar mentalmente la forma de la 'y' con la 'v' de 'vertical'.

La sentencia

PLOT *coordenada x, coordenada y*

«tiñe» el pixel correspondiente a esas coordenadas, de forma que este programa

```
10 PLOT INT (RND*256), INT (RND*176): INPUT a$: GO TO 10
```

dibuja un punto en una posición aleatoria cada vez que usted pulsa INTRO.

Veamos un programa más interesante; traza un gráfico de la función SIN (una onda sinusoidal) para valores del arco comprendidos entre 0 y 2π:

```
10 FOR n=0 TO 255
20 PLOT n,88+80*SIN (n/128*PI)
30 NEXT n
```

El siguiente programa traza un gráfico de SQR (parte de una parábola) entre 0 y 4:

```
10 FOR n=0 TO 255
20 PLOT n,80*SQR (n/64)
30 NEXT n
```

Observe que las coordenadas que describen la posición de los pixels son distintas de las que utilizábamos para dar el número de fila y de columna en la cláusula AT. Quizá le resulte útil el diagrama de la Parte 15 de este capítulo a la hora de elegir las coordenadas de los pixels y los números de fila y columna.

PLOT realiza los dibujos punto a punto. También podemos dibujar rectas, circunferencias y trozos de circunferencia utilizando las sentencias DRAW y CIRCLE.

La sentencia DRAW dibuja una recta; su forma es:

```
DRAW x,y
```

El punto de partida de la recta es el pixel en el que se detuvo la última sentencia PLOT, DRAW o CIRCLE (que es lo que llamamos *posición actual* del cursor gráfico; RUN, CLEAR, CLS y NEW la restablecen en el extremo inferior izquierdo, punto 0,0). El punto final de la recta se encuentra a x pixels a la derecha de la posición actual y a y por arriba. La sentencia DRAW determina por sí misma la longitud y la dirección de la recta, pero no su punto de partida.

Experimente con unas cuantas órdenes PLOT y DRAW; por ejemplo,

```
PLOT 0,100: DRAW 80,-35
PLOT 90,150: DRAW 80,-35
```

Observe que los números de la sentencia DRAW pueden ser negativos, porque representan desplazamientos con respecto a una posición, mientras que los de PLOT tienen que ser positivos, porque representan coordenadas absolutas.

También podemos dibujar en color, aunque hay que tener presente que los colores cubren siempre la totalidad de la celda de un carácter y no pueden ser especificados para pixels individuales. Cuando dibujamos un pixel, éste se tiñe del color de la tinta, y el resto de la celda en que se encuentra el pixel toma el mismo color, como demuestra el siguiente programa:

```
10 BORDER 0: PAPER 0: INK 7: CLS: REM tiñe la pantalla de negro
20 LET x1=0: LET y1=0: REM comienzo de la recta
30 LET c=1: REM para color de tinta, comenzando con azul
40 LET x2=INT (RND*256): LET y2=INT (RND*176): REM final aleatorio de la
   recta
50 DRAW INK c;x2-x1,y2-y1
60 LET x1=x2: LET y1=y2: REM la siguiente recta comienza donde acabó la anterior
70 LET c=c+1: IF c=8 THEN LET c=1: REM otro color
80 GO TO 40
```

Las rectas parecen hacerse más amplias a medida que avanza el programa. Esto se debe a que cada recta cambia los colores de todas las celdas que atraviesa. Observe que podemos introducir las cláusulas PAPER, INK, FLASH, BRIGHT, INVERSE y OVER en las sentencias PLOT y DRAW, lo mismo que hacíamos en PRINT e INPUT. Estas cláusulas van entre la palabra clave y las coordenadas y terminan en coma o en punto y coma.

Una característica adicional de DRAW es que podemos usarla para dibujar fragmentos de circunferencias en vez de rectas. La sentencia necesita entonces un número más para especificar el ángulo del trozo de circunferencia. La forma es

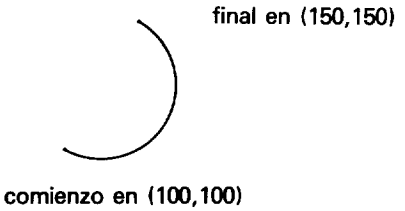
DRAW *x,y,a*

donde *x* e *y* especifican, al igual que antes, el punto final de la línea, y donde *a* especifica el arco (en número de radianes). Si *a* es positivo, el giro se realiza hacia la izquierda; si es negativo, hacia la derecha. Otra forma de entender el significado de *a* es considerar que indica qué fracción de una circunferencia completa se va a dibujar (una circunferencia completa tiene 2π radianes). Por ejemplo, si $a=\pi$, obtendremos una semicircunferencia; si $a=0.5\pi$, un cuarto de circunferencia; y así sucesivamente.

Supongamos que $a=\pi$. Entonces, cualesquiera que sean los valores de *x* e *y*, siempre obtendremos una circunferencia. Pruebe el programa:

```
10 PLOT 100,100: DRAW 50,50,PI
```

que dibuja lo siguiente:



El dibujo comienza en dirección sudeste, pero, en el momento en que se detiene, ya va en dirección noroeste. Entre el rumbo inicial y el final ha girado 180 grados, o π radianes, que es el valor de a.

Ejecute el programa varias veces reemplazando PI por otras expresiones; por ejemplo: −PI, PI/2, 3*PI/2, PI/4, 1, 0, etc.

La última sentencia de esta sección es CIRCLE, que dibuja una circunferencia completa. La circunferencia se especifica dando las coordenadas del centro y el radio. La forma de CIRCLE es:

CIRCLE *coordenada x,coordenada y,radio*

Al igual que sucedía con PLOT y DRAW, al principio de CIRCLE podemos poner cláusulas que controlen el color y los otros atributos del dibujo.

La función POINT averigua si un pixel tiene el color de la tinta o el del papel. Sus dos argumentos son las coordenadas del pixel (que deben ir entre paréntesis). El resultado es 0 si el pixel tiene el color del papel, o 1 si es del color de la tinta. Pruebe lo siguiente:

```
CLS : PRINT POINT (0,0): PLOT 0,0: PRINT POINT (0,0)
```

Escriba:

```
PAPER 7: INK 0
```

e investiguemos ahora cómo funcionan INVERSE y OVER dentro de una sentencia PLOT. Ambas afectan sólo al pixel especificado en PLOT, no al resto de la celda de carácter. Normalmente estas cláusulas están desactivadas (a 0) en las sentencias PLOT, así que sólo es necesario mencionarlas cuando se desee activarlas (ponerlas a 1).

He aquí una lista de las posibilidades:

PLOT;

Ésta es la forma usual. Dibuja un punto con la tinta; es decir, tiñe el pixel con el color de la tinta.

PLOT INVERSE 1;

Dibuja un punto con el 'borrador de tinta'; es decir, hace que el pixel muestre el color del papel.

PLOT OVER 1;

Intercambia el color del pixel con el que tuviera antes, de modo que si antes era el de la tinta lo convierte en el del papel, y vice versa.

PLOT INVERSE 1; OVER 1;

Esto deja el pixel exactamente como estaba antes, pero cambia la posición actual; puede servir, pues, para mover el cursor gráfico.

Para ver otro ejemplo de aplicación de la sentencia OVER, llene la pantalla de texto escrito en negro sobre blanco y después haga

```
PLOT 0,0: DRAW OVER 1;255,175
```

Esto dibuja una recta bastante aceptable, aunque con puntos en blanco cada vez que se tropieza con algo escrito previamente. Ahora dé otra vez exactamente la misma orden: la recta desaparece sin dejar rastro alguno; ésta es la gran ventaja de OVER 1. Si hubiéramos dibujado la recta con

```
PLOT 0,0: DRAW 255,175
```

y la hubiéramos borrado con

```
PLOT 0,0: DRAW INVERSE 1;255,175
```

habríamos borrado también parte del texto.

Ahora pruebe

```
PLOT 0,0: DRAW OVER 1;250,175
```

y trate de borrar con

```
DRAW OVER 1;-250,-175
```

Esto no funciona demasiado bien, y es que los pixels por los que pasa la recta en su camino de vuelta no son exactamente los mismos que los que había recorrido a la ida. Por consiguiente, para borrar una recta hay que recorrerla en el mismo sentido en que se la dibujó.

Una manera de obtener colores especiales es mezclar dos colores normales en un cuadrado, usando para ello un gráfico definible por el usuario. Pruebe el siguiente programa:

```
1000 FOR n=0 TO 6 STEP 2
1010 POKE USR "a"+n, BIN 01010101: POKE USR "a"+n+1, BIN 10101010
1020 NEXT n
```

que genera un gráfico definido por el usuario con el diseño de un tablero de ajedrez. Si ahora escribimos este carácter (pulse GRAF y luego A) en tinta roja sobre papel amarillo, obtendremos un naranja bastante aceptable.

Ejercicios

1. Experimente con las cláusulas PAPER, INK, FLASH y BRIGHT en una sentencia PLOT. Éstas son las partes que afectan a la totalidad de la celda de carácter en la que se encuentra el pixel. Normalmente, es como si la sentencia PLOT hubiera empezado con

   ```
   PLOT PAPER 8; FLASH 8; BRIGHT 8;  etc.
   ```

   y sólo se altera el color de la tinta de una celda de carácter cuando se dibuja algo en ella, pero usted puede cambiar esta situación si lo desea.

   Tenga cuidado especialmente cuando use colores con INVERSE 1, ya que esta cláusula hace que el pixel muestre el color del papel, y puede cambiar el color de la tinta, el cual pudiera no ser el que usted esperaba.

2. Pruebe a dibujar circunferencias usando SIN y COS con el siguiente programa (si ha leído la Parte 10, trate de entender cómo funciona):

   ```
   10 FOR n=0 TO 2*PI STEP PI/180
   20 PLOT 100+80*COS n,87+80*SIN n
   30 NEXT n
   40 CIRCLE 150,87,80
   ```

   Como puede ver, la sentencia CIRCLE es mucho más rápida, pero menos precisa.

3. Pruebe lo siguiente:

   ```
   CIRCLE 100,87,80: DRAW 50,50
   ```

   De esto se deduce que la sentencia CIRCLE deja el cursor en una posición bastante imprecisa (siempre se trata de algún punto a media altura en el lado derecho de la circunferencia). Normalmente habrá que ejecutar una sentencia PLOT a continuación de una CIRCLE antes de seguir dibujando.

Parte 18
Movimiento

Temas tratados:

PAUSE, INKEY$, PEEK

Con frecuencia se necesita poder controlar el tiempo que dura la ejecución de un programa. Para este fin se dispone de la sentencia PAUSE.

PAUSE *n*

detiene el programa y mantiene la imagen durante *n* barridos del cuadro (a razón de 50 barridos por segundo en Europa, 60 en EE.UU.). El valor de *n* puede llegar hasta 65535, número que produce una pausa de 22 minutos. Si $n=0$, la pausa es de duración infinita.

La pausa se abandona en cualquier momento pulsando una tecla.

El siguiente programa mueve el segundero de un reloj:

```
 10 REM primero dibujamos la esfera del reloj
 20 FOR n=1 TO 12
 30 PRINT AT 10-10*COS (n/6*PI),16+10*SIN (n/6*PI);n
 40 NEXT n
 50 REM ahora ponemos en marcha el reloj
 60 FOR t=0 TO 200000: REM t es el tiempo en segundos
 70 LET a=t/30*PI: REM a es el ángulo del segundero en radianes
 80 LET sx=80*SIN a: LET sy=80*COS a
200 PLOT 128,88: DRAW OVER 1;sx,sy: REM dibuja el segundero
210 PAUSE 42
220 PLOT 128,88: DRAW OVER 1;sx,sy: REM borra el segundero
400 NEXT t
```

El reloj deja de funcionar al cabo de unas 55.5 horas, debido a la línea 60, pero usted puede darle más cuerda fácilmente. Observe que el control del tiempo lo lleva la línea 210. Parecería natural que PAUSE 50 realizase el batido de segundos, pero también hay que tener en cuenta el tiempo que tarda el ordenador en llevar a cabo las restantes instrucciones del programa. La forma de ajustar el parámetro de PAUSE es comparar el funcionamiento del programa con un cronómetro hasta dar con el valor correcto. (El ajuste no puede ser demasiado perfecto; una diferencia de un barrido más o menos representa un error del 2%, es decir, aproximadamente media hora al día.)

Hay otro método mucho más exacto para medir el tiempo, basado en el contenido de ciertas posiciones de memoria. Los datos almacenados en la memoria se leen con la función PEEK. La Parte 25 de este capítulo explica con detalle el significado exacto de las posiciones de memoria que vamos a leer. La expresión que necesitamos es:

```
(65536*PEEK 23674+256*PEEK 23673+PEEK 23672)/50
```

que da el número de segundos transcurridos desde que se encendió o reinicializó el ordenador (hasta alrededor de tres días y 21 horas, porque después de ese tiempo vuelve a 0).

El siguiente programa es una versión revisada del programa anterior en la que utilizamos esa expresión:

```
 10 REM primero dibujamos la esfera del reloj
 20 FOR n=1 TO 12
 30 PRINT AT 10-10*COS (n/6*PI),16+10*SIN (n/6*PI);n
 40 NEXT n
 50 DEF FN t()=INT ((65536*PEEK 23674+256*PEEK 23673+PEEK 23672)/50):
    REM número de segundos desde la reinicialización
100 REM ahora ponemos en marcha el reloj
110 LET t1=FN t()
120 LET a=t1/30*PI: REM a es el ángulo del segundero en radianes
130 LET sx=72*SIN a: LET sy=72*COS a
140 PLOT 131,91: DRAW OVER 1;sx,sy: REM dibujar segundero
200 LET t=FN t()
210 IF t<=t1 THEN GO TO 200: REM esperar hasta que llegue el momento de
    moverlo
220 PLOT 131,91: DRAW OVER 1;sx,sy: REM borrar segundero
230 LET t1=t: GO TO 120
```

El reloj interno en el que se basa este método tiene una precisión de alrededor del 0.01% (aproximadamente 10 segundos por día), a condición de que lo único que haga el ordenador sea ejecutar el programa. Sin embargo, cuando se ejecuta la sentencia BEEP (descrita en la Parte 19 de este capítulo) o se trabaja con el magnetófono, la impresora o cualquier otro periférico, el reloj interno se detiene temporalmente y por lo tanto se atrasa.

Los números PEEK 23674, PEEK 23673 y PEEK 23672 están almacenados en el ordenador y constituyen un contador de cincuentavos de segundo. Crecen gradualmente de 0 a 255, y, al llegar a 255, vuelven a cero.

El que se incrementa más a menudo es PEEK 23672 (una unidad cada 1/50 segundos). Cuando llega a 255, el siguiente incremento lo lleva a 0, y al mismo tiempo suma 1 a PEEK 23673. Cuando PEEK 23673 pasa de 255 a 0 (cada 256/50 segundos), esto a su vez suma 1 a PEEK 23674. Esta explicación debería ser suficiente para que usted comprenda cómo funciona la expresión anterior.

Suponga que nuestros tres números son 0 (PEEK 23674), 255 (PEEK 23673) y 255 (PEEK 23672). Esto significa que hace unos 21 minutos que se encendió el ordenador. El valor de la expresión será

```
(65536*0+255+255)/50=1310.7
```

Pero hay un peligro oculto: la próxima vez que se incremente el contador en 1/50 segundos, los tres números cambiarán a 1, 0 y 0. Alguna vez ocurrirá esto en el preciso momento en que el ordenador está calculando la expresión; entonces el +2 leerá el número 0 en PEEK 23674, pero los otros dos valores cambiarán a cero antes de que el programa los haya leído. El valor de la expresión será en ese caso:

```
(65536*0+256*0+0)/50=0
```

que, evidentemente, no es correcta.

Una manera sencilla de evitar este problema es evaluar la expresión dos veces seguidas y tomar la respuesta mayor. Si usted observa cuidadosamente el programa anterior, verá que hace esto implícitamente.

Veamos un truco para aplicar esta regla. Defina las funciones:

```
10 DEF FN m(x,y)=(x+y+ABS (x-y))/2: REM el mayor de x e y
20 DEF FN u()=(65536*PEEK 23674+256*PEEK 23673+PEEK 23672)/50:
   REM tiempo (puede ser incorecto)
30 DEF FN t()=FN m(FN u(), FN u()): REM tiempo (correcto)
```

Podemos cambiar los tres números del contador para que den el tiempo real en vez del tiempo que lleva el ordenador encendido. Por ejemplo, para poner el contador en hora a las 10 de la mañana, observemos que 10 horas son $10\times60\times60\times50=1800000$ cincuentavos de segundo y que

```
1800000=65536*27+256*119+64
```

Por eso debemos escribir los números 27, 119 y 64 en las tres posiciones de memoria, y eso se hace con la siguiente orden:

```
POKE 23674,27: POKE 23673,119: POKE 23672,64
```

En los países en los que la frecuencia de la red de energía eléctrica sea de 60 Hz, el número 50 que aparece en todos estos programas debe ser cambiado por 60.

La función INKEY$ (que no tiene argumento) lee el teclado. Si en el momento de evaluar INKEY$ tenemos pulsada una tecla sola (o combinada con [MAYUSC]), el resultado es el carácter que esa tecla da normalmente; de lo contrario, el resultado es la cadena vacía.

Pruebe este programa, que funciona como una máquina de escribir:

```
10 IF INKEY$<>"" THEN GO TO 10
20 IF INKEY$="" THEN GO TO 20
30 PRINT INKEY$;
40 GO TO 10
```

La línea 10 espera hasta que usted retira sus dedos del teclado; la línea 20 espera hasta que pulsa una nueva tecla.

Recuerde que, a diferencia de INPUT, INKEY$ no le espera, de modo que no tiene que pulsar [INTRO]. Por otra parte, si usted no escribe absolutamente nada, ha perdido su oportunidad.

Ejercicios

1. ¿Qué ocurre si omitimos la línea 10 del programa de la máquina de escribir?

2. Otra forma de usar INKEY$ consiste en combinarla con PAUSE, como en este segundo programa de máquina escribir:

   ```
   10 PAUSE 0
   20 PRINT INKEY$;
   30 GO TO 10
   ```

   ¿Por qué es esencial, para que esto funcione, que la pausa no acabe si ya se encuentra usted pulsando una tecla cuando comienza?

3. Adapte el programa del segundero de forma que también muestre el minutero y la manecilla de las horas, redibujándolas una vez por minuto. Si se siente usted ambicioso, haga que produzca algún efecto adicional cada cuarto de hora, quizá el repique de las campanas del 'Big Ben' usando PLAY (orden descrita en la Parte 19 de este capítulo).

Parte 19
Sonido

Temas tratados:

BEEP, PLAY

Como usted ya habrá observado, el +2 puede producir diversos ruidos. Para obtener la mejor calidad de sonido posible, es importante que el televisor esté perfectamente sintonizado (véase capítulo 2). Si en lugar de un televisor está usted usando un monitor (que no reproducirá el sonido del +2), observe que puede tomar del zócalo **SOUND** de la parte posterior del ordenador una señal de sonido, la cual puede ser enviada a los altavoces o auriculares a través de un amplificador de audio. Tenga en cuenta, no obstante que los auriculares *no* pueden ser conectados directamente al ordenador.

Las conexiones del zócalo **SOUND** están descritas en el capítulo 10.

El conocimiento de la terminología musical ayuda a extraer el mayor partido posible de la capacidad sonora del +2.

Nota. En los ejemplos que siguen, es importante que usted escriba las expresiones literales exactamente como nosotros las mostramos, en mayúsculas y minúsculas; así, por ejemplo, "ga" no es lo mismo que "Ga", "gA" o "GA".

Escriba esta orden (de momento no se preocupe por lo que significa):

```
PLAY "ga"
```

Suenan dos notas, la segunda ligeramente más alta (aguda) que la primera. La diferencia entre las notas se llama *tono*. Ahora pruebe

```
PLAY "g$a"
```

De nuevo han sonado dos notas; la primera era la misma que en el ejemplo anterior, pero el salto hasta la segunda fue menor. Si no ha notado la diferencia, pruebe otra vez el primer ejemplo y luego el segundo.

La diferencia entre las dos notas del segundo ejemplo es un *semitono*.

Cuando se haya familiarizado con los semitonos, pruebe esto:

```
PLAY "gD"
```

Esta diferencia se llama *quinta* y aparece frecuentemente en todos los tipos de música. Con ese ejemplo todavía en sus oídos, escriba:

```
PLAY "gG"
```

A pesar de que ahora la diferencia entre las notas ha sido mucho mayor que en el caso de la quinta, de algún modo las notas sonaron bastante más parecidas. Esta diferencia se llama una octava, y es el punto a partir del cual la música empieza a 'repetirse a sí misma'. No se preocupe demasiado por esto; basta con que recuerde cómo suena una octava.

Hay dos formas de producir música y sonidos con el +2. La más elemental es la orden, un poco espartana, BEEP. Su forma es:

BEEP *duración,altura*

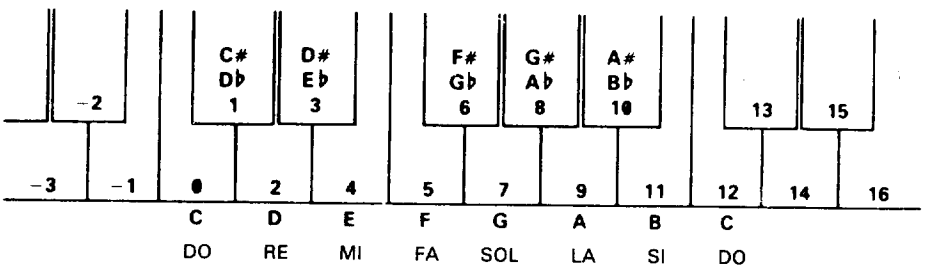
donde, como de costumbre, *duración* y *altura* son expresiones numéricas. La duración se da en segundos; la altura, en número de semitonos por encima de la nota DO medio; los números negativos representan las notas por debajo de la DO medio.

NOTA

Para representar las notas musicales, el +2 utiliza la nomenclatura anglosajona. La equivalencia entre ésta y la española es como sigue:

C DO
D RE
E MI
F FA
G SOL
A LA
B SI

El siguiente diagrama muestra los valores del parámetro *altura* para todas las notas de una octava en el piano.



Por consiguiente, para producir la nota LA que está por encima de la DO medio, con duración de medio segundo, debemos dar la orden:

```
BEEP 0.5,9
```

Para interpretar una escala (por ejemplo, DO mayor) se necesita un programa completo (aunque corto):

```
10 FOR f=1 to 8
20 READ nota
30 BEEP 0.5,nota
40 NEXT f
50 DATA 0,2,4,5,7,9,11,12
```

Para obtener notas más altas o más bajas que las mostradas en el diagrama, se debe sumar o restar 12 por cada octava que quiera subir o bajar.

La orden BEEP ha sido incluida más que nada por compatiblidad con modelos más antiguos del Spectrum, aunque también puede ser útil para generar efectos sonoros muy cortos o rápidos. Para cualquier programa nuevo que usted desarrolle, la segunda forma de producir sonido, basada en la orden PLAY, es, con mucho, la preferible.

PLAY es mucho más flexible que BEEP; puede interpretar hasta tres voces en armonía con todo tipo de efectos, y ofrece un sonido de calidad muy superior. Además, es bastante más fácil de usar. Por ejemplo, para interpretar la nota LA posterior a DO medio durante medio segundo, dé la orden

```
PLAY "a"
```

Para interpretar la escala de DO mayor basta con

```
PLAY "cdefgabC"
```

Observe que la última C de este ejemplo es mayúscula. Este hecho indica a la orden PLAY que debe interpretar esta nota una octava más arriba que la representada por la c minúscula. A propósito, una *escala* es el nombre que se da al conjunto de las notas que hay en una octava. La escala del ejemplo anterior se llama la 'escala de Do mayor' porque es el conjunto de notas entre un DO y el siguiente. ¿Por qué 'mayor'? Hay dos tipos básicos de escala: mayor y menor. Esta terminología es sólo una forma abreviada de describir dos conjuntos diferentes. Sólo por si le interesa, la escala de DO menor suena así:

```
PLAY "cd$efg$a$bC"
```

Cuando una nota va precedida de $, suena un semitono más baja (*bemol*); si el prefijo es #, suena un semitono más alta (*sostenido*). La orden PLAY cubre 9 octavas; se puede especificar la octava deseada escribiendo la letra O mayúscula seguida de un número.

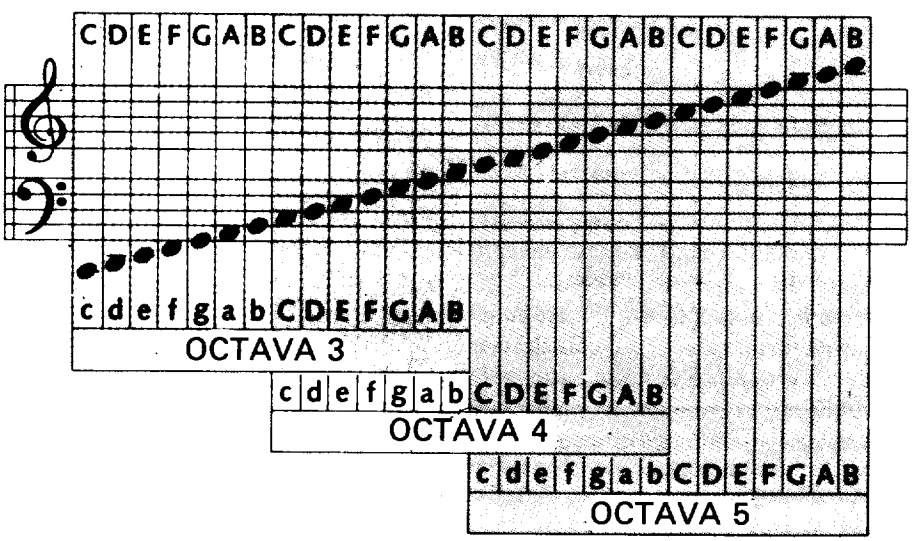
Pruebe este programa:

```
10 LET o$="O5"
20 LET n$="DECcg"
30 LET a$=o$+n$
40 PLAY a$
```

Hay unas cuantas novedades en este programa. En primer lugar, PLAY funciona igual de bien con una variable literal que con una constante. En otras palabras, suponiendo que a$ ha sido definida con anterioridad, PLAY a$ produce el mismo efecto que PLAY "O5DECcg". De hecho, el uso de variables en sentencias PLAY tiene algunas otras ventajas, y nosotros las emplearemos de aquí en adelante.

Observe también que la cadena a$ se ha formado por concatenación de dos cadenas más pequeñas: o$ y n$. Aunque esto no representa gran ventaja cuando las cadenas son así de cortas, el hecho es que PLAY puede manejar cadenas de una longitud de muchos miles de notas, y la única forma práctica de crear y editar esas cadenas en BASIC es combinar cadenas más pequeñas.

Ahora ejecute el programa anterior. Edite la línea 10 para poner "O7" en lugar de "O5" y ejecútelo de nuevo. Pruébelo también con "O2". Si no especifica número de octava en una cadena determinada, el +2 toma por defecto la octava 5. El siguiente diagrama da las notas y números de octava que corresponden a la 'escala musical bien temperada'.



Hay un solapamiento considerable; por ejemplo, "O3D" es lo mismo que "O4d". Gracias a esto se puede escribir melodías sin tener que especificar demasiados cambios de

escala. Por razones técnicas, algunas de las notas de las octavas más bajas (0 y 1) no son muy exactas, pero el ordenador trata de desafinar lo menos posible.

PLAY puede producir notas de diferente duración. Edite el programa anterior de forma que la línea 10 sea ahora:

```
10 LET o$="2"
```

y ejecútelo. Después pruebe con otros valores de o$, entre "1" y "9". La longitud de nota se puede especificar en cualquier lugar de la cadena incluyendo un número entre el "1" y el "9"; la nueva duración queda en vigor hasta que PLAY encuentra otro número más adelante. Cada una de estas nueve duraciones de nota tiene un nombre musical específico. La tabla siguiente da los nombres y notaciones musicales:

Número	Nombre de la nota	Símbolo musical
1	simicorchea	
2	semicorchea con puntillo	
3	corchea	
4	corchea con puntillo	
5	negra	
6	negra con puntillo	
7	blanca	
8	blanca con puntillo	
9	redonda	

PLAY también puede producir *tresillos*, que son grupos de tres notas interpretadas en el tiempo de dos. A diferencia de las duraciones de nota sencilla, el número de tresillo sólo afecta a las tres notas siguientes; después de ellas continúa en vigor el anterior número de duración. Los números de tresillo son los siguientes:

Número	Nombre de la nota	Símbolo musical
10	tresillo de semicorcheas	
11	tresillo de corcheas	
12	tresillo de negras	

Además, PLAY sabe callarse a tiempo. Al periodo de tiempo durante el cual no se interpreta ninguna nota se lo llama *silencio*. Un silencio se representa por &; su longitud es la que esté en vigor para las notas. Edite las líneas 10 y 20 y cámbielas por

```
10 LET o$="O4"
20 LET n$="DEC&cg"
```

Dos notas ejecutadas juntas forman una *ligadura*. En PLAY las ligaduras se especifican mediande un signo de subrayado. Por ejemplo, una DO negra y una DO blanca ligadas se representan por 5_7c (el segundo de los dos números especifica la duración para las notas siguientes).

Hay ocasiones en las que se presenta alguna ambigüedad. Supongamos que una pieza musical necesita la octava 6 y una duración de nota de 2; entonces

```
10 LET o$="O62"
```

parece una buena baza, pero no lo es. El ordenador encontrará la O y tratará de leer el número siguiente. Como lo que encuentra es el 62, se detiene y emite el mensaje de error n Out of range ('Fuera del margen'). Para casos como éste disponemos de una 'nota auxiliar', llamada N, que sólo sirve como separador. Así, en la línea 10 deberíamos poner:

```
10 LET o$="O6N2"
```

El volumen es ajustable entre 0 (mínimo) y 15 (máximo) escribiendo el número detrás de la letra V. En la práctica, si no se utiliza un amplificador, sólo resultan útiles los números del 10 al 15, ya que los del 0 al 9 son demasiado suaves. Como ya hemos mencionado, BEEP produce un sonido más intenso que un canal de PLAY; pero si se hace sonar PLAY en los tres canales a volumen 15, la intensidad será igual a la de BEEP.

El manejo de varios canales es muy sencillo; basta con especificar en PLAY varias listas de notas, separadas con comas. Pruebe este nuevo programa:

```
10 LET a$="O4cCcCgGgG"
20 LET b$="O6CaCe$bd$bD"
30 PLAY a$,b$
```

En general, no hay diferencia entre los tres canales, y cualquier cadena de notas puede ser reproducida en cualquier canal. La velocidad global de la música, el *tempo*, debe estar en la cadena asignada al canal A (la primera que se especifica tras PLAY), pues de lo contrario será ignorada. Para especificar el tempo en número de notas (negras) por minuto, se escribe la letra T seguida de un número comprendido entre 60 y 240. El valor estándar es 120, que equivale a dos negras por segundo. Modifique el programa anterior de esta forma:

```
 5 LET t$="T120"
10 LET a$=t$+"O4cCcCgGgG"
20 LET b$="O6CaCe$bd$bD"
30 PLAY a$,b$
```

y ejecútelo varias veces poniendo en la línea 5 tempos diferentes.

Un fenómeno muy frecuente en la música es la repetición de un grupo de notas. Cualquier parte de una cadena puede ser repetida escribiéndola entre paréntesis. Así, si cambiamos la línea 10 por:

```
10 LET a$=t$+"O4(cC)(gG)"
```

el efecto es el mismo que obteníamos antes. Si ponemos el paréntesis de cerrar sin un paréntesis de abrir anterior que le corresponda, se repite indefinidamente todo el tramo previo de la cadena, desde el principio hasta el paréntesis. Esto es aprovechable en los efectos rítmicos y líneas de bajo. Para demostrarlo, pruebe lo siguiente (tendrá que usar BREAK para detener el sonido):

```
PLAY "O4N2cdefgfed)"
```

y después

```
PLAY "O4N2cd(efgf)ed)"
```

Si usted prepara una línea de bajo que se repita indefinidamente y después la utiliza para acompañar una melodía, sería interesante que el acompañamiento terminase al mismo tiempo que la melodía. Afortunadamente, en PLAY hay un mecanismo que nos permite lograrlo: si PLAY se encuentra la letra H en cualquiera de las cadenas que está interpretando, detiene todos los sonidos inmediatamente. Ejecute el siguiente programa (de nuevo tendrá que pulsar BREAK para detenerlo):

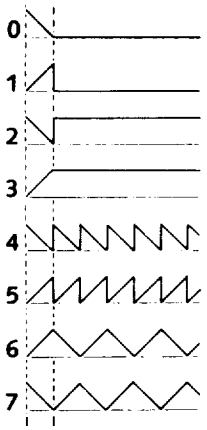
```
10 LET a$="cegbdfaC"
20 LET b$="O4cC)"
30 PLAY a$,b$
```

Ahora modifique la línea 10 de la forma siguiente:

```
10 LET a$="cegbdfaCH"
```

y ejecútelo de nuevo.

Hasta aquí sólo hemos usado notas que comienzan y terminan en el mismo nivel de volumen. El +2 puede alterar el volumen de una nota mientras está interpretándola, de forma que, por ejemplo, la nota puede ser intensa al principio y decaer lentamente (como las del piano). Para controlar este efecto se utiliza la letra W (de *waveform*, 'forma de onda') seguida de un número entre 0 y 7, junto con una U para cada canal al que se quiera aplicar el efecto. Si en un canal determinado se ha especificado volumen con V, no responderá a la U. El siguiente diagrama muestra la forma en que varía el volumen a lo largo del tiempo para los diveros valores del parámetro de V.



0 caída simple y fin
1 ataque simple y fin
2 caída simple y sostenimiento
3 ataque simple y sostenimiento
4 caída repetida
5 ataque repetido
6 ataque-caída repetidos
7 caída-ataque repetidos

El siguiente programa interpreta la misma nota con cada uno de estos valores. Trate de «escuchar» las formas ilustradas en el diagrama.

```
10 LET a$="UX1000W0C&W1C&W2C&W3C&W4C&W5C&W6C&W7C"
20 PLAY a$
```

La U activa los efectos y la W selecciona la forma de onda. La cláusula X1000 establece cuánto tiempo deben durar los efectos (su parámetro debe estar entre 0 y 65535). Si no se incluye la X, el +2 escogerá el valor más largo. Las formas que llegan a estabilizarse (1 a 3 en la tabla anterior) funcionan mejor con valores de X en torno a 1000, mientras que las periódicas (4 a 7) son más eficaces con valores pequeños, tales como 300. Pruebe diversos valores de X en el programa anterior para hacerse una idea de cómo funciona cada uno.

La orden PLAY no está limitada a las notas puramente musicales. Hay también tres generadores de 'ruido blanco' (ruido blanco es, más o menos, el ruido que produce la radio en FM o el televisor cuando no están sintonizados). Cualquiera de los tres canales puede interpretar notas, ruido blanco o una mezcla de ambas cosas. Para seleccionar esta mezcla se especifica la letra M seguida de un número (del 1 al 63). El significado del número es el que se indica en la siguiente tabla.

	Canales de tono			Canales de ruido		
	A	B	C	A	B	C
Número	1	2	4	8	16	32

Para obtener el número que se debe especificar tras M, se toma nota de los números que corresponden a los efectos que se desea activar y luego se los suma. Por ejemplo, si quisiéramos ruido en al canal A, tono en el B y ambas cosas en el C, tendríamos que sumar

8, 2, 4 y 32 para obtener 46 (el orden de los canales es el orden en que ponemos las cadenas tras la orden PLAY). Los mejores efectos se obtienen en el canal A; no se prive de experimentar.

A estas alturas ya estará usted escribiendo sinfonías. Sin embargo, cuando las cosas se complican, puede ser difícil recordar qué tramo concreto de una cadena es responsable de cierta parte de la música. Para aliviar este problema, la cadena puede incluir comentarios escritos entre signos de admiración !. Por ejemplo,

```
1098 LET z$=z$+"CDcE3Ge4_6f! fin del compás 75 !egeA"
```

La orden PLAY sencillamente se salta los comentarios y los ignora.

Si usted dispone de un instrumento musical electrónico con MIDI, el +2 puede controlarlo con la orden PLAY. Hasta ocho canales de música pueden ser enviados a los sintetizadores, tambores y secuenciadores. La orden PLAY se construye exactamente como hemos explicado en esta sección, con la única diferencia de que cada cadena debe incluir una Y seguida de un número (entre 1 y 16). Este número controla a qué canal son asignados los datos de la música. Se puede usar hasta ocho cadenas; las tres primeras seguirán siendo interpretadas a través del televisor, como antes, por lo que puede ser conveniente bajar el volumen del aparato. También se puede enviar códigos de programación del MIDI a través de la orden PLAY, usando para ello una Z seguida del número de código. Las velocidades (intensidad sonora) se calculan y envían como 8 veces el valor de V (así, V6 enviará el número 48 como velocidad).

Para concluir esta sección, vamos a dar una lista de los parámetros que pueden ser incluidos en las cadenas de PLAY, junto con los valores que pueden tomar:

Cadena	Función
a–g A–G	Especifican el tono de la nota, dentro de la octava actual
$	Especifica que la nota siguiente debe ser convertida en bemol
#	Especifica que la nota que sigue debe ser convertida en sostenido
O*n*	Especifica el número de octava (*n* entre 0 y 8)
1–12	Especifica la duración de las notas
&	Especifica un silencio
_	Especifica una ligadura
N	Separador entre números
V*n*	Especifica el volumen de las notas (*n* entre 0 y 15)
W*n*	Especifica el efecto de variación de volumen (*n* entre 0 y 7)
U	Introduce el efecto de variación de volumen en una cadena
X*n*	Especifica la duración del efecto de variación de volumen (*n* entre 0 y 65535)

Cadena	Función
T*n*	Especifica el tempo de la música (*n* entre 60 y 240)
()	Los paréntesis especifican que la frase escrita entre ellos debe ser repetida
!!	Los signos de admiración especifican que el comentario escrito entre ellos debe ser ignorado
H	Interrumpe la generación de sonido
M*n*	Especifica qué canal o canales, musicales o de ruido, debe usar PLAY (*n* entre 1 y 63)
Y*n*	Especifica qué canal MIDI se debe usar (*n* entre 1 y 16)
Z*n*	Especifica un código de programación del MIDI (*n* es el número del código)

Parte 20
Operaciones con el magnetófono

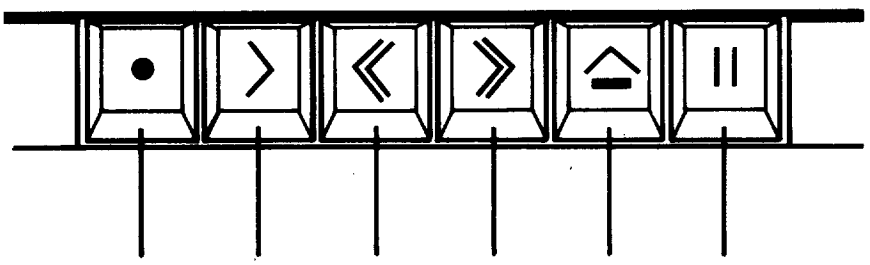
Temas tratados:

LOAD, SAVE, VERIFY, MERGE

El procedimiento básico de uso del magnetófono para cargar programas está explicado en los capítulos 3 y 4.

También podemos usar el magnetófono para almacenar (grabar) programas en una cinta, de forma que podamos volver a cargarlos en el ordenador siempre que deseemos usarlos (de no ser por esto, tendríamos que volver a escribir el programa cada vez que lo necesitásemos).

Antes de nada, familiarícese con las seis teclas de control del magnetófono:



Para estudiar cómo se graba un programa en el magnetófono, empiece por transcribir el programa que vimos al final de la Parte 16 (el que llenaba la pantalla con cuadrados de colores):

```
10 POKE 22527+RND*704, RND*127
20 GO TO 10
```

Éste es el programa que vamos a grabar. Cualquier cinta de cassette estándar servirá, aunque son preferibles las de bajo nivel de ruido.

Escriba lo siguiente:

```
SAVE "cuadrados"
```

"cuadrados" es justamente el nombre que utilizamos para 'etiquetar' el programa en el momento de almacenarlo en la cinta. Los nombres de los programas pueden constar de hasta 10 caracteres.

El ordenador mostrará en la pantalla el siguiente mensaje:

```
Pulse REC y PLAY, y luego tecla
```

Primero llevaremos a cabo una grabación simulada para que usted pueda ver lo que ocurrirá cuando más tarde grabemos el programa realmente. Esta vez, pues, no pulse los botones de grabación y marcha, sino sólo una tecla del ordenador (por ejemplo, INTRO), y observe el borde de la pantalla. Verá las siguientes pautas de rayas de colores:

Cinco segundos de rayas de color rojo y cyan desplazándose lentamente hacia arriba, seguidas por una cortísima irrupción de rayas azules y amarillas.

Una pausa corta.

Dos segundos de rayas rojas y cyan de nuevo, seguidas por una corta irrupción de rayas azules y amarillas.

Al tiempo que aparecen las rayas en la pantalla, también se puede oír el «sonido» de los datos a través del altavoz del televisor.

Siga probando la orden SAVE anterior (sin poner en marcha el magnetófono) hasta que pueda reconocer esas pautas. Lo que está sucediendo en realidad es que la información está siendo grabada en dos *bloques*, y ambos tienen una «cabecera» (que corresponde a las rayas rojas y cyan) a la que sigue la información propiamente dicha (que corresponde a las rayas azules y amarillas). El primero es un bloque preliminar que contiene el nombre y alguna otra información sobre el programa; el segundo es el programa con sus variables. La pausa entre ellos es sólo un hueco, sin ningún otro significado.

Ahora grabemos realmente el programa en la cinta:

1. Haga avanzar o retroceder la cinta hasta una zona que esté libre o en la que usted haya decidido grabar.
2. Escriba:

```
SAVE "cuadrados"
```

3. Obedezca el mensaje Pulse REC y PLAY, y luego tecla.
4. Observe que la pantalla se comporta como vimos antes. Cuando el +2 haya terminado de grabar (informe 0 OK), pulse el botón de parada del magnetófono.

Después de grabar *con éxito* un programa, ya se puede apagar tranquilamente el ordenador, o reinicializarlo, o bien comenzar un programa nuevo (NEW), sabiendo que podrá volver a cargar el programa siempre que lo necesite. No obstante, antes de borrarlo de

la memoria del ordenador, debe comprobar que el proceso de grabación ha funcionado correctamente; puede comparar la señal que ha quedado grabada en la cinta con el programa que todavía está en la memoria usando la orden VERIFY:

1. Rebobine la cinta hasta poco antes antes del punto en el que inició la grabación del programa.
2. Escriba:

   VERIFY "cuadrados"

   El borde alternará entre los colores rojo y cyan hasta que el +2 encuentre el programa especificado; después mostrará la misma pauta que apareció cuando se grabó el programa. Durante la pausa entre los bloques, aparecerá el mensaje Program: cuadrados. (Cuando el +2 está buscando algo en la cinta, muestra en la pantalla el nombre de todo lo que encuentra.) Si, después de haber aparecido la pauta, el ordenador emite el mensaje 0 OK, eso quiere decir que el programa está bien grabado, y entonces usted puede saltarse los cinco párrafos siguientes. De lo contrario, ha habido algún problema; siga este procedimiento para averiguar en qué consiste:

   Si el nombre del programa no ha aparecido en la pantalla, puede ser que el programa no quedara grabado o que, si quedó bien grabado, el ordenador no haya podido leerlo. Tenemos que averiguar cuál de estas dos cosas ha ocurrido. Para ver si la grabación es correcta, rebobine la cinta hasta antes de donde empezó la grabación del programa y vuelva a pasarla mientras escucha el altavoz del televisor. La cabecera (rojo y cyan) debe producir una nota clara y persistente, de tono alto; la parte de la información (azul y amarillo) producirá un chirrido menos agradable.

   Si no oye estos ruidos, es probable que el programa no quedara grabado. Asegúrese de que no está tratando de grabar el programa en la guía de plástico del principio de la cinta. Cuando lo haya hecho, intente otra vez grabar el programa.

   Si puede oir los sonidos descritos, entonces la grabación (SAVE) es correcta y el problema está en la lectura.

   Quizá haya escrito mal el nombre del programa al grabarlo; en tal caso, cuando el +2 encuentra el programa muestra en pantalla el nombre incorrecto. También puede haber escrito mal el nombre tras la orden VERIFY, en cuyo caso el ordenador ignorará el programa correctamente grabado y continuará buscando el nombre erróneo, provocando destellos de rojo y cyan según avanza.

   Si realmente hay un error en la cinta, el +2 mostrará el mensaje R Tape loading error ('Error de carga de la cinta'), lo cual significa en este caso que ha fracasado la operación de verificación. Un pequeño defecto de la cinta misma (que podría ser casi inaudible en una grabación musical) puede causar estragos en un programa de ordenador. Trate de grabar el programa de nuevo, tal vez en una parte diferente de la cinta.

Ahora supongamos que usted ha grabado el programa y lo ha verificado con éxito. La carga del programa en la memoria es similar a la verificación, con la diferencia de que ahora la orden es:

```
LOAD "cuadrados"
```

(en lugar de VERIFY "cuadrados").

Puesto que la verificación tuvo éxito, la carga no debería dar ningún problema.

LOAD borra el programa (y las variables) que pudiera haber en la memoria cuando carga el nuevo programa desde la cinta.

Una vez cargado el programa, se puede dar la orden RUN para ejecutarlo.

Como hemos dicho en los capítulos 3 y 4, usted puede comprar programas comerciales pregrabados en cinta. Tales programas deben haber sido escritos especialmente para los ordenadores ZX Spectrum (es decir, para el Spectrum, el Spectrum +, el Spectrum 128 o el Spectrum +2). Cada marca y modelo de ordenador tiene una forma diferente de almacenar programas, así que con este ordenador no es posible leer las cintas grabadas por o para otros.

Si en la misma cara de la cinta hay varios programas, cada uno tendrá un nombre. En la orden LOAD se puede especificar qué programa se desea cargar; por ejemplo, si el que queremos cargar se llama 'helicoptero', podemos escribir:

```
LOAD "helicoptero"
```

La orden LOAD "" significa que se debe cargar el primer programa que el ordenador encuentre en la cinta. Esto puede ser muy útil si usted no recuerda el nombre con el que grabó el programa.

La opción Carg. cinta del menú de presentación tiene el mismo efecto que LOAD "", y es mucho más fácil y rápida de usar (basta con encender el ordenador y pulsar INTRO).

Como ya hemos dicho, LOAD borra del ordenador el programa y las variables antiguos siempre que se carga los nuevos desde la cinta. No obstante, hay otra orden, MERGE, que es similar a LOAD, pero que sólo borra una línea o una variable antigua si hay una nueva con el mismo número o nombre. Escriba el programa 'dados' de la Parte 11 de este capítulo y grábelo en la cinta con el nombre de dados. Después transcriba y ejecute el siguiente programa:

```
 1 PRINT 1
 2 PRINT 2
10 PRINT 10
20 LET x=20
```

Rebobine la cinta hasta poco antes de donde comenzó la grabación del programa dados y luego dé la orden

```
MERGE "dados"
```

Siga el mismo procedimiento que si estuviera cargando (LOAD) el programa. Si ahora lista (LIST) el programa, verá que las líneas 1 y 2 han sobrevivido y que las líneas 10 y 20 han sido sustituidas por las del programa dados. La variable x también se conserva (compruébelo con PRINT x).

Ya hemos visto las formas más sencillas de las cuatro órdenes que trabajan en colaboración con el magnetófono:

SAVE • Almacena en la cinta el programa y las variables.

VERIFY • Compara el programa y las variables leídos en la cinta con los de la memoria del ordenador.

LOAD • Borra el programa y todas las variables del ordenador y los sustituye por los nuevos que ha leído en la cinta.

MERGE • Similar a LOAD, pero **no** borra las líneas ni las variables del programa antiguo a menos que tenga que hacerlo (por tener el mismo número o nombre en los dos programas).

En todas estas órdenes la palabra clave va seguida por una cadena. Para la orden SAVE, la cadena consiste en el nombre con el cual se almacena el programa en la cinta, mientras que para las otras tres órdenes la cadena indica al ordenador qué programa buscar. Cuando el ordenador está buscando, muestra en pantalla el nombre de los programa que encuentra en la cinta. Sin embargo, hay un par de peculiaridades en todo esto:

VERIFY, LOAD y MERGE pueden ir seguidas de una cadena vacía "" como nombre de programa; entonces el ordenador no se preocupa por el nombre, sino que toma el primer programa que encuentra.

Hay una variante de SAVE que tiene la forma:

SAVE *cadena* LINE *número*

Un programa grabado con esta orden se almacena de forma tal que, cuando es leído por LOAD (pero no por MERGE), salta automáticamente a la línea especificada por el número dado, ejecutándose por tanto a sí mismo.

Si usted carga un programa que no es de ejecución automática (usando la opción Carg. cinta del menú de presentación), después de cargarlo tendrá que seleccionar la opción 128 BASIC para poder ejecutarlo o editarlo.

Hasta aquí, los únicos tipos de información que hemos almacenado en cinta han sido programas junto con sus variables. Hay otros dos tipos más, llamados *matrices* y *bytes*.

Podemos grabar matrices en cinta incluyendo la cláusula DATA en la sentencia SAVE:

SAVE *cadena* DATA *nombre-de-matriz*()

donde *cadena* es el nombre con que la información quedará grabada en la cinta y funciona exactamente igual que cuando grabamos un programa (o simples bytes).

El *nombre-de-matriz* especifica la matriz que usted quiere grabar, así que tiene que ser una sola letra (o una letra seguida de $). No olvide poner los paréntesis () tras el nombre de la matriz.

Distinga bien los papeles que desempeñan *cadena* y *nombre-de-matriz*. Si, por ejemplo, hacemos:

```
SAVE "Vargas" DATA b()
```

entonces SAVE toma la matriz b del ordenador y la almacena en la cinta con el nombre de Vargas.

Cuando luego demos la orden:

```
VERIFY "Vargas" DATA b()
```

el ordenador buscará una matriz numérica almacenada en la cinta con el nombre de Vargas. Cuando la encuentre, escribirá en la pantalla el mensaje Number array: Vargas ('Matriz numérica') y la comparará con la matriz b que está en la memoria del ordenador.

La orden:

```
LOAD "Vargas" DATA b()
```

encuentra la matriz en la cinta y después (si hay espacio para ella en el ordenador) borra la matriz que pueda existir con el nombre de b y carga la nueva desde la cinta, llamándola b.

No se puede usar MERGE para cargar matrices.

También podemos grabar matrices de caracteres (literales), exactamente de la misma forma que las numéricas. Cuando el ordenador está buscando en la cinta y encuentra una matriz literal, escribe en la pantalla Character array: ('Matriz de caracteres') y a continuación el nombre. Cuando carga una matriz literal, el ordenador borra no sólo la matriz literal, sino también la variable literal que se encuentre en la memoria con el mismo nombre.

El almacenamiento de bytes se aplica a bloques de información cualesquiera, con independencia de la naturaleza de esa información (podría ser una imagen de la pantalla, o quizá algunos gráficos definidos por el usuario, etc.). Este tipo de grabación se realiza incluyendo la cláusula CODE en la sentencia SAVE; por ejemplo,

```
SAVE "imagen" CODE 16384,6912
```

La unidad de almacenamiento en memoria es el *byte* (un número entero comprendido entre 0 y 255); cada byte de la memoria está identificado por una *dirección* (que es un número comprendido entre 0 y 65535). El primer número que se pone después de CODE es la dirección del primer byte que debe ser grabado en la cinta; el segundo número es la cantidad de bytes que deben ser grabados. En nuestro caso, 16384 es la dirección del primer byte del fichero (que contiene la imagen de pantalla); 6912 es la cantidad de bytes que hay en él. Pruebe la anterior orden SAVE. (No tiene por qué grabar los bytes precisamente con el nombre imagen; nosotros hemos elegido este nombre para que nos recuerde cuál es la naturaleza de la información grabada.)

Para cargar esta información se da la orden

```
LOAD "imagen" CODE
```

También podemos poner números detrás de CODE:

```
LOAD nombre CODE comienzo,longitud
```

Aquí *longitud* se utiliza como medida de seguridad. Cuando el ordenador ha encontrado en la cinta los bytes con el *nombre* solicitado, comprueba la *longitud* y, en caso de que haya más, sólo carga los bytes especificados; de esta forma se previene el riesgo de sobreescritura accidental de una zona de la memoria que interese preservar. Además, en tal caso, el ordenador emite el mensaje R Tape loading error.

Si omitimos la *longitud*, el ordenador lee los bytes que encuentra, sin importarle cuántos son.

El parámetro *comienzo* especifica la dirección de memoria en la que debe ser cargado el primer byte, que puede ser diferente de la dirección desde la que fue grabado. No obstante, si ambas direcciones son iguales, se puede omitir el parámetro *comienzo* en la sentencia LOAD.

CODE 16384,6912 es un área de memoria (imagen de pantalla) tan útil, que BASIC dispone de una función especial, SCREEN$, para representarla. Así, para grabar y cargar la pantalla se puede usar las órdenes:

```
SAVE "imagen" SCREEN$
```

y

```
LOAD "imagen" SCREEN$
```

Éste es uno de los pocos casos en los que VERIFY no funciona. En efecto, puesto que VERIFY escribe los nombres de los ficheros que encuentra en la cinta, altera la imagen de la pantalla, y entonces ésta ya no puede ser igual a la grabada en la cinta.

Cualquier operación que podamos hacer en la cinta con SAVE, LOAD o MERGE puede ser realizada también con el *disco de silicio* (que está incorporado al +2 y que actúa como si fuese una cinta, con un par de órdenes adicionales). La diferencia entre la cinta

y el disco de silicio consiste en que éste tiene una capacidad de almacenamiento de unos 64K y un tiempo de acceso muy pequeño; la desventaja es que el contenido del disco de silicio se pierde cuando se apaga o reinicializa el ordenador (sin embargo, «sobrevive» a la orden NEW). Todas las órdenes se utilizan exactamente igual que con el magnetófono, pero intercalando un signo de admiración ! entre la palabra clave y la cadena asociada. Así, en lugar de grabar en la cinta con

```
SAVE "cuadrados"
```

podemos grabar en el disco de silicio con

```
SAVE ! "cuadrados"
```

Hay dos órdenes adicionales que pueden ser aplicadas al disco de silicio. La primera es

```
CAT !
```

que le da una lista de todos los programas o ficheros de datos almacenados en el disco.

La segunda orden es

```
ERASE ! "nombre-de-fichero"
```

que borra el programa o fichero de datos especificado.

Quizá la aplicación más obvia del disco de silicio sea el almacenamiento de porciones de un programa de BASIC para mezclarlas (con MERGE !) sucesivamente con un programa más pequeño. Esto hace posible tener en la memoria un programa de cerca de 90K (para hacer esto la estructura del programa tiene que estar bien definida).

Una de las aplicaciones más atractivas del disco de silicio es la *animación* de imágenes. Un programa de BASIC, de por sí relativamente lento, puede definir una serie de imágenes y almacenarlas en la memoria. Éstas pueden ser luego transferidas a la pantalla a gran velocidad. El siguiente programa da una idea de lo que se puede lograr con esta técnica (sin duda, usted puede hacer algo mejor):

```
 10 INK 5: PAPER 0: BORDER 0: CLS
 20 FOR f=1 TO 10
 30 CIRCLE f*20,150,f
 40 SAVE ! "balon"+STR$(f) CODE 16384,2048
 50 CLS
 60 NEXT f
 70 FOR f=1 TO 10
 80 LOAD ! "balon"+STR$(f) CODE
 90 NEXT f
100 BEEP 0.01, 0.01
110 FOR f=9 TO 2 STEP -1
120 LOAD ! "balon"+STR$(f) CODE
```

```
130 NEXT f
140 BEEP 0.01, 0.01
150 GO TO 70
160 REM utilice GO TO 160 para borrar las imágenes del disco
170 FOR f=10 TO 1 STEP -1
180 ERASE ! "balon"+STR$(f)
190 NEXT f
```

En la línea 40, los dos números que siguen a CODE son la dirección de memoria donde empieza la pantalla y la longitud del tercio superior de ésta. Al grabar y cargar solamente ese trozo de la pantalla se mejora la velocidad global. Hemos incluido la rutina que va de la línea 160 a la 190 por si usted quiere detener el programa, modificar la porción que dibuja las circunferencias y grabar las nuevas imágenes. Así pues, para borrar las imágenes del disco de silicio, dé la orden GO TO 160. (Es conveniente borrar los ficheros «hacia atrás», de forma que el último fichero grabado será el primero en ser borrado; esto facilita el trabajo al ordenador y hace que el proceso sea mucho más rápido.)

Para concluir esta sección, vamos a dar un resumen de las cuatro instrucciones de control del magnetófono:

Notas

- El parámetro *nombre* representa una expresión literal y se refiere al nombre con el que la información se graba en la cinta. Debe consistir en caracteres ASCII, de los cuales el ordenador sólo utiliza los 10 primeros.
- Hay cuatro clases de información que pueden ser almacenadas en cinta o en el disco de silicio: programa y variables (juntos), matrices numéricas, matrices literales y bytes.
- Cuando VERIFY, LOAD y MERGE están buscando información en la cinta con un nombre dado y de una clase determinada, el ordenador muestra en la pantalla la *clase* y el *nombre* de toda la información que encuentra. La clase la indica mediante Program: (programa), Number array: (matriz numérica), Character array: (matriz literal) o Bytes: (bytes). Si el *nombre* es la cadena vacía (""), el ordenador toma el primer bloque de información (de la clase correcta) sin tener en cuenta el nombre.
- En lo que sigue, los corchetes [] indican que ! es opcional. Cuando se incluye esta cláusula, la orden se refiere al disco de silicio.

SAVE

1. Programa y variables:

   SAVE [!] *nombre* LINE *numero-de-línea*

   graba el programa y las variables de tal forma que LOAD implica automáticamente un GO TO *numero-de-línea*.

2. Bytes:

   SAVE [!] *nombre* CODE *comienzo,longitud*

   graba el número de bytes especificado por *longitud*, empezando por la dirección *comienzo*.

   Observe que

   SAVE [!] *nombre* SCREEN$

   es equivalente a

   SAVE [!] *nombre* CODE 16384,6912

   y graba la imagen de la pantalla.

3. Matrices:

   SAVE [!] *nombre* DATA *letra*()

   o

   SAVE [!] *nombre* DATA *letra*$()

   graba la matriz numérica cuyo nombre es *letra*, o la matriz literal cuyo nombre es *letra*$.

VERIFY

1. Programa y variables:

   VERIFY *nombre*

   lee el programa y las variables grabados en la cinta con el *nombre* especificado y los compara con los que están en la memoria.

2. Bytes:

   VERIFY *nombre* CODE *comienzo,longitud*

   Si los bytes grabados como *nombre* no son más de *longitud*, entonces compara los bytes de la cinta con los de la memoria, empezando por la dirección *comienzo*.

   VERIFY *nombre* CODE

compara los bytes grabados en la cinta como *nombre* con los de la memoria, empezando por la dirección *comienzo*.

3. Matrices:

   VERIFY *nombre* DATA *letra*()

   o

   VERIFY *nombre* DATA *letra*$()

   compara la matriz numérica cuyo nombre es *letra*, o la matriz literal cuyo nombre es *letra*$, con la matriz *letra* o *letra*$ de la memoria.

LOAD

1. Programa y variables:

   LOAD [!] *nombre*

   borra el programa y las variables antiguos y carga desde la cinta el programa y las variables grabados con el *nombre* especificado. Si el programa había sido grabado con SAVE *nombre* LINE *número-de-línea*, entonces LOAD ejecuta un GO TO *número-de-línea* automático en cuanto termina de cargar el programa.

   Si la carga no tiene éxito, ni el programa ni las variables antiguos son borrados.

2. Bytes:

   LOAD [!] *nombre* CODE *comienzo,longitud*

   Si los bytes grabados como *nombre* no son más de *longitud*, los carga en la memoria desde la cinta, empezando en la dirección *comienzo* y escribiendo sobre lo que hubiese previamente en esa zona de la memoria.

   LOAD [!] *nombre* CODE *comienzo*

   carga incondicionalmente en la memoria desde la cinta los bytes grabados con el *nombre* especificado, empezando en la dirección *comienzo* y escribiendo sobre lo que hubiese previamente en esa zona de la memoria.

   LOAD [!] *nombre* CODE

   carga en la memoria desde la cinta los bytes grabados con el *nombre* especificado, empezando en la dirección a partir de la cual fue grabado el primer byte, y escribiendo sobre los bytes que estaban antes en esa zona de la memoria.

3. Matrices:

   LOAD [!] *nombre* DATA *letra*()

   o

   LOAD [!] *nombre* DATA *letra*$()

   borra la matriz numérica que pudiera haber en la memoria con el nombre de *letra*, o la matriz literal llamada *letra*$, y crea una nueva, con ese nombre, a base de los datos leídos en la cinta.

MERGE

1. Programa y variables:

   MERGE [!] *nombre*

   mezcla el programa grabado con el *nombre* especificado con el que ya está en la memoria, sobreescribiendo sólo las líneas y las variables del programa antiguo cuyos números o nombres existan también en el nuevo.

2. Bytes:
   No es posible.

3. Matrices:
   No es posible.

Ejercicio

1. Practique la grabación, carga y mezcla de programas y de ficheros de datos, tanto en cinta como en disco de silicio.

Parte 21
Operaciones de la impresora

Temas tratados:

LPRINT, LLIST, COPY

El +2 tiene incorporada una puerta serie y el software necesario para controlar una impresora. Estos recursos sólo pueden ser utilizados en modo 128 BASIC.

La impresora debe tener un interfaz de tipo serie RS232; además, para que pueda reproducir las imágenes de pantalla, debe tener *modo gráfico de imagen de bits de cuádruple densidad compatible Epson.*

Asegúrese de que tiene el cable correcto para conectar la impresora con el +2; en caso de duda, consulte a su distribuidor.

Para lograr que el +2 y la impresora se comuniquen el uno con la otra, ambos deben tener la misma *velocidad de transmisión*, que es, como su nombre indica, la velocidad a la que se transfieren los datos del ordenador a la impresora. Aunque la velocidad de transmisión pueda ser seleccionada en la impresora, probablemente será más fácil cambiarla, si es necesario, en el ordenador. En algún lugar del manual de la impresora estará especificada la velocidad de transmisión; averigüe cuál es y selecciónela en +2 con la orden:

FORMAT "p"; *velocidad de transmisión*

(No será necesario hacer esto si la impresora funciona normalmente a 9600 baudios, pues ésta es la velocidad que el +2 supone por defecto.)

Una vez preparada la conexión, ya podemos empezar a usar las tres órdenes de BASIC que controlan la impresora. Las dos primeras, LPRINT y LLIST, son análogas a PRINT y LIST, salvo que dirigen la salida a la impresora en lugar de al televisor. La opción Imprimir del menú de edición de 128 BASIC produce el mismo efecto que LLIST, pero la hemos incluido para proporcionar una forma más directa y fácil de obtener un listado.

Pruebe este programa:

```
10 PRINT "Este programa ""
20 LLIST
30 LPRINT ""imprime el juego de caracteres: ""
40 FOR n=32 TO 255
50 LPRINT CHR$ n;
60 NEXT n
```

Es importante observar que LPRINT y LLIST tienen buen cuidado de filtrar todos los códigos de color (y sus parámetros) que encuentran antes de imprimir o listar cualquier cosa. Tales códigos de color son una «reliquia» del Spectrum de 48K; cuando se los incluía en una cadena ajustaban INK (tinta), PAPER (papel), etc. En general las impresoras utilizan esos códigos para cosas completamente diferentes, como activar o desactivar cursiva, subrayado, etc., así que sería muy peligroso enviarles estos códigos de color y esperar que no ocurriese ningún incidente. Como un efecto secundario de esto, es imposible (desde BASIC) controlar cualquier función especial en una impresora que utilice *secuencias de escape* (carácter 27) o códigos de control similares.

La tercera instrucción, COPY, imprime una copia de la pantalla del televisor. Para ver una primera demostración, entre en la pantalla pequeña, dé la orden LIST para tener en la pantalla algo que imprimir y después escriba:

```
COPY
```

La orden COPY tarda de 15 a 30 segundos en prepararse para enviar datos a la impresora, así que no se inquiete si no parece que ocurra nada inmediatamente. Finalmente obtendrá otro listado del programa en la impresora, pero esta vez se parecerá bastante a lo que había en la pantalla. Si, en cambio, todo lo que produce COPY en la impresora es un montón de caracteres aleatorios, es probable que la impresora no sea del todo compatible.

En cualquier momento se puede detener la impresión pulsando la tecla [BREAK]. Algunas impresoras tienen lo que se conoce por el nombre de *tampón*, una memoria en la que almacenan el texto antes de imprimirlo. Si su impresora tiene tampón, al pulsar [BREAK] no se detendrá inmediatamente, a pesar de que el +2 dejará de enviarle datos en ese mismo momento.

Si usted intenta ejecutar una orden LLIST, LPRINT o COPY cuando la impresora no está conectada, el +2 se pondrá a esperar pacientemente que la impresora (inexistente) le diga que está preparada. Pulsando [BREAK] volverá a despertar al ordenador.

Pruebe este programa:

```
10 FOR n=31 TO 0 STEP -1
20 PRINT AT 31-n,n; CHR$ (CODE "0"+n);
30 NEXT n
```

Verá una serie de caracteres bajando diagonalmente desde el extremo superior derecho hasta el borde inferior de la pantalla, y en ese momento el programa preguntará si quiere desplazar la pantalla.

Ahora cambie el AT 31-n,n de la línea 20 por TAB n. El programa producirá exactamente el mismo efecto que antes.

Ahora cambie PRINT en la línea 20 por LPRINT. Esta vez el ordenador no se para a preguntar scroll?, pues esto sólo ocurre cuando la salida está siendo dirigida a la pantalla.

Ahora vuelva a sustituir TAB n por AT 31−n,n manteniendo LPRINT. Esta vez obtendrá solamente una única línea de símbolos. La causa de esta diferencia es que el resultado de LPRINT no se imprime inmediatamente, sino que se almacena en un tampón hasta que se haya acumulado el equivalente a una línea de impresora, o bien hasta que alguna otra cosa provoque el vaciado del tampón. Así pues, la impresión sólo tiene lugar:

1. Cuando se llena el tampón.
2. Tras una sentencia LPRINT que no acabe en coma ni en punto y coma.
3. Cuando una coma, apóstrofo o cláusula TAB obligue a cambiar de línea.
4. Al final de un programa, si ha dejado algo sin imprimir.
5. En algunas impresoras, cuando se pulsa el botón de 'fuera de línea'.

El punto 3 explica por qué nuestro programa con TAB funciona así. En cuanto a AT, el número de fila es ignorado y la posición de LPRINT (igual que la de PRINT) se desplaza a la columna especificada. Un elemento AT no puede hacer nunca que la línea sea enviada a la impresora.

Ejercicio

1. Imprima un gráfico de la función seno: ejecute el primer programa de la Parte 17 de este capítulo y luego dé la orden COPY.